УДК 519.6

С.Г. Григорьев,
Н.М. Добровольский,
А.Р. Есаян

Параметрические функции для обработки гнездовых массивов

В статье предложена концепция обработки гнездовых массивов *M* пользовательскими функциями *F* с реализацией их в системе *PTC Mathcad Prime*. Суть концепции состоит в том, что создаваемая функция *F* одним из своих аргументов должна иметь вспомогательную встроенную или пользовательскую функцию *f*, работающую с простыми массивами, числами или строками. Фиксируя *f* при обращениях к *F*, мы будем решать конкретную задачу по обработке *M*. При этом если *f* является встроенной функцией, то обращаться к *F* можно непосредственно, а если нет — то предварительно требуется создать пользовательскую функцию *f*.

Ключевые слова: информатика; гнездовые массивы; информационные технологии; система инженерных и научных вычислений *PTC Mathcad Prime*.

Напомним, что простые матрицы — это матрицы, элементами которых могут быть числа или строки. Гнездовые матрицы определяются рекурсивно как матрицы, элементами которых могут быть числа, строки и матрицы, причем во вложенных матрицах элементами снова могут быть числа, строки и матрицы. Поскольку матрицы являются двумерными или одномерными массивами, то вместо слов «*матрица*», «*простая матрица*» и «*гнездовая матрица*» часто используются слова «*массив*», «*простой массив*» и «*гнездовой массив*».

В статьях [1–5] для работы с гнездовыми массивами предложена серия рекурсивных функций, реализованных на языке программирования системы инженерных и научных вычислений *PTC Mathcad Prime*. В [2; 3] речь идет о функциях общего назначения; в [4; 5] — о функциях, позволяющих проводить «векторные» операции, то есть действия сразу над каждым из элементов массива; в [1] — о функциях, осуществляющих простой или обобщенный поиск элементов в гнездовых массивах и их замещение.

Подчеркнем еще раз, что создавать придется не функцию для обработки гнездовых массивов, а лишь функцию f для обработки простых массивов, чисел или строк. Таким образом, связь создаваемых функций F с некоторыми вспомогательными функциями-уточнителями f позволяет решать по F не конкретные задачи, а некоторые совокупности задач, то есть массовые проблемы. Фактически описанные функции F являются параметрическими с аргументом-параметром, являющимся встроенной или пользовательской функцией f, уточняющей постановку задачи.

Гнездовой массив удобно интерпретировать деревом, корнем которого является сам массив, а от него идут дуги к элементам — скалярам, строкам и гнездовым массивам. От массивов снова идут дуги к их элементам — скалярам, строкам и массивам и т. д. Листьями подобного дерева являются скаляры или строки — конечные элементы, не имеющие последующих ссылок. Исходя из этого, для гнездовых массивов используют многие термины, применяемые для деревьев: такие как листья, уровни вложенности элементов, высота (максимальная глубина вложенности элементов) и т. п.

Постановка задач. В этом пункте приведены формулировки всех решаемых задач.

Задачи А. «L (sa) = ?» Пусть L (sa) — встроенная или пользовательская логическая функция одной переменной, определенная на простых массивах sa со скалярными и (или) строковыми элементами и принимающая значения 1 (истина) и 0 (ложь). Требуется построить функцию, по которой в гнездовом массиве ma находится:

– $A1$. Один из простых подмассивов sa, удовлетворяющий условию L (sa) = 1. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*».

– $A2$. Вектор всех простых подмассивов sa, удовлетворяющих условию L (sa) = 1. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*».

Приведем примеры конкретных функций L, уточняющих (конкретизирующих) постановку задач $A1$ и $A2$. Пусть b — действительное число, sa — простой числовой массив. Задавать L будем утверждениями относительно sa:

1–5. L (sa) — в sa существуют элементы, равные b ($< b$, $<= b$, $> b$, $>= b$).

6–10. L (sa) — в sa все элементы равны b ($< b$, $<= b$, $> b$, $>= b$).

11–15. L (sa) — в sa сумма элементов равна b ($< b$, $<= b$, $> b$, $>= b$).

16. L (sa) — в sa есть элементы кратные b.

17. L (sa) — столбцы sa упорядочены по неубыванию.

18. L (sa) — столбцы sa упорядочены по невозрастанию.

19. L (sa) — строки sa упорядочены по неубыванию.

20. L (sa) — строки sa упорядочены по невозрастанию.

21. L (sa) — ранг sa является полным ($rank = min$ ($rows$, $cols$)).

22. L (sa) — след sa равен b.

....

Ясно, что этот список можно расширять по своему усмотрению.

Задачи B. $P\,(sa, x) = ?$ Пусть $P\,(sa, x)$ — встроенная или пользовательская логическая функция двух переменных, определенная на простых массивах sa и x со скалярными и (или) строковыми элементами и принимающая значения 1 (истина) и 0 (ложь). Требуется построить функцию, по которой в гнездовом массиве ma находится:

*B*1. Простой подмассив sa такой, что для любых простых подмассивов x из ma выполняется условие $P\,(sa, x) = 1$. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*».

*B*2. Вектор всех простых подмассивов sa таких, что для любых простых подмассивов x из ma выполняется условие $P\,(sa, x) = 1$. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*».

Приведем примеры конкретных функций P, уточняющих (конкретизирующих) постановку задач *B*1 и *B*2:

1. $P\,(sa, x)$ — количество элементов в sa равно количеству элементов в x.
2. $P\,(sa, x)$ — сумма элементов sa равна сумме элементов x.
3. $P\,(sa, x)$ — произведение элементов sa равно произведению элементов x.
4. $P\,(sa, x)$ — минимальный элемент sa равен минимальному элементу x.
5. $P\,(sa, x)$ — максимальный элемент sa равен максимальному элементу x.
6. $P\,(sa, x)$ — минимальный по модулю элемент sa равен минимальному по модулю элементу x.
7. $P\,(sa, x)$ — максимальный по модулю элемент sa равен максимальному по модулю элементу x.
8. $P\,(sa, x)$ — сумма квадратов элементов sa равна сумме квадратов элементов x.
9. $P\,(sa, x)$ — ранг матрицы sa равен рангу x.
10. $P\,(sa, x)$ — определитель матрицы sa равен определителю x.
11. $P\,(sa, x)$ — след матрицы sa равен следу x.
12. $P\,(sa, x)$ — максимальное собственное значение sa равно максимальному собственному значению x.

13–60. Следующие 4 группы функций $P(sa,x)$ получаются заменой в 1–12 слова «*равно*» («*равен*», «*равна*») на «*меньше*», «*больше*», «*меньше или равно*» и «*больше или равно*».

... ...

Представленный список можно пополнять по своему усмотрению.

Задачи C. $P\,(sa, x) = ?$ Пусть $P\,(sa, x)$ — встроенная или пользовательская логическая функция двух переменных, определенная на простых массивах sa и x со скалярными и/или строковыми элементами и принимающая значения 1 (истина) и 0 (ложь). Требуется построить функцию, по которой в гнездовом массиве ma находится (пока все как в задаче *B*):

*C*1. Простой подмассив sa такой, что существует простой подмассив x из ma (x^1 sa), на котором выполняется условие $P\,(sa, x) = 1$. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*».

*C*2. Вектор всех простых подмассивов *sa* таких, что существует простой подмассив *x* из *ma* (x^1 *sa*), на котором выполняется условие *P* (*sa*, *x*) = 1. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*» (в выводимом векторе могут быть повторяющиеся подмассивы).

*C*3. Вектор всех различных простых подмассивов *sa* таких, что существует простой подмассив *x* из *ma* (x^1 *sa*), на котором выполняется условие *P* (*sa*, *x*) = 1. При отсутствии таких подмассивов должно быть выведено сообщение «*нет*» (в выводимом векторе не может быть повторяющихся массивов).

Примерами конкретных функций *P*, уточняющих постановку задач *C*1–*C*3, могут быть те же самые функции, которые предлагались для задач *B*.

I. Вспомогательные функции. Для решения сформулированных выше 7-ми задач *A*1–*A*2, *B*1–*B*2 и *C*1–*C*3 нам потребуются несколько вспомогательных функций. Все они приведены в разделе I фрагмента 2. Перечислим их:

- рекурсивная функция *hei* вычисления высоты, то есть глубины вложенности, массива *ma*;
- функция *leaves* проверки, что массив *ma* является простым;
- рекурсивная функция *vecsim* создания по массиву *ma* вектора всех его простых подмассивов;
- функция *vecsims* создания по массиву *ma* вектора всех его различных простых подмассивов (повторяющиеся подмассивы входят в вектор по одному разу).

Все вспомогательные функции протестированы на массивах, приведенных в этом же разделе.

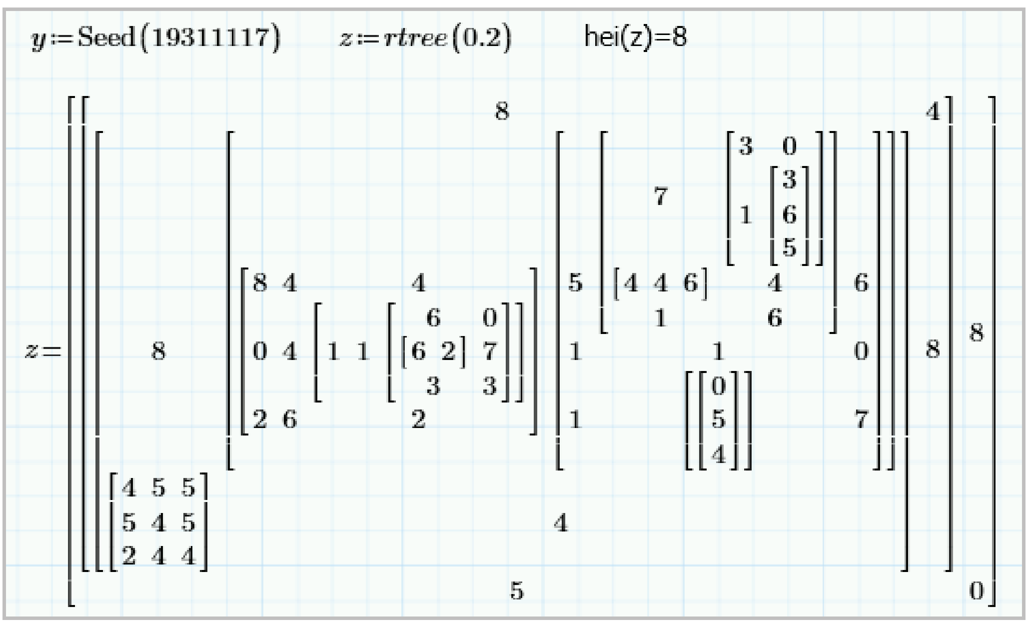
II. Генерирование случайных гнездовых массивов. Для того чтобы можно было проводить тестирование создаваемых функций на большом количестве гнездовых массивов, проще всего организовать их автоматическое генерирование. Сделать это можно программным путем. В разделе II фрагмента 2 приведены две функции *rtree* и *setree*, справляющиеся с поставленной задачей. Они генерируют гнездовые массивы со случайным количеством строк и столбцов, случайной глубины вложенности и случайными элементами (при равномерном распределении). Кратко опишем их:

- Рекурсивная функция *rtree*(*p*). Функция *rtree* генерирует случайные массивы с целыми неотрицательными случайными элементами из диапазона 0...8. Случайные размеры массива и его подмассивов устанавливаются равными $(n + 1) \times (m + 1)$, где $n =$ *floor* (*rnd* (3)) и $m =$ *floor* (*rnd* (3)) (*rnd* (*k*)) возвращает случайное число *x*, такое, что 0 £ $x < k$, $k > 0$). Случайная глубина вложенности массива задается выбранным способом формирования его элементов. Если сгенерированное по *rnd* (1) число оказывается меньше *p* ($0 < p < 1$), то реализуется уход в глубину, то есть рекурсивный вызов *rtree*, иначе формируется конечный элемент-лист. Случайное целочисленное значение листа из диапазона 0…8 определяется присваиваниями *floor* (*rnd* (9)). Из сказанного ясно, как можно при необходимости организовать изменение случайных параметров для *rtree*: размеров массива и его подмассивов, диапазона значений элементов и высоту массива.

- Головная функция *setree* (*p*). Прямые обращения к *rtree* могут приводить к ошибкам вычислений, связанных с переполнением стека рекурсивных вызовов или попытками вывести в документ сгенерированную громоздкую гнездовую матрицу. В *setree* происходит обращение к *rtree* с обработкой возможных упомянутых ошибок.

Замечание. По *setree* выводятся случайные гнездовые матрицы с глубиной вложенности не более 5. Если требуется получить гнездовую матрицу с большей глубиной вложенности, можно напрямую использовать функцию *rtree*, отказавшись от обработки ошибок, или изменить в *setree* значение в правой части условия цикла *while*. Ниже в качестве примера на фрагменте 1 показана сгенерированная по *rtree* случайная гнездовая матрица *z* с глубиной вложенности 8. Функция *Seed*(*x*) используется в следующих целях. По ней устанавливается начальное значение *x* ($1 \le x \le 2147483647$) ядра генератора случайных чисел, и она же возвращает последнее сгенерированное число. При повторных выполнениях команд «*y*:=*Seed*(*x*) *z*:=*rtree*(*p*)» мы будем получать ту же самую гнездовую матрицу (т. е. совпадающую с первоначальной), что весьма полезно при отладке программ.

Фрагмент 1



III. Функции для тестирования программ. Все создаваемые функции решения задач *A*1–*A*2, *B*1–*B*2 и *C*1–*C*3 мы собираемся тестировать на большом количестве автоматически генерируемых гнездовых массивов. Для этого необходимо сформировать соответствующие функции тестирования. Во фрагменте 2 приведены функции *testa* и *testftime*, реализующие такие тесты.

Фрагмент 2 (1–3)

I. Вспомогательные функции
II. Генерирование случайных гнездовых массивов
III. Функции для тестирования программ

Создаваемые функции будем проверять на:

a) гнездовых массивах

$$a0 := \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} & 7 \\ 41 & \begin{bmatrix} \begin{bmatrix} 2 \\ 6 \end{bmatrix} & \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} & \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$a1 := [5] \qquad a2 := [[[5]]] \qquad a3 := [[1]\ [2]] \qquad a4 := [[1\ [[1]\ [2]]]]$$

$$a5 := \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \end{bmatrix};$$

b) гнездовых массивах, приведенных в соответствующем разделе;

c) генерируемых случайных гнездовых массивах.

I. Вспомогательные функции

I*a*. *Рекурсивная. функция вычисления высоты (глубины вложенности) массива ma.*

```
hei(ma) := | h ← 0
           | if IsArray(ma)
           |   | for b ∈ ma
           |   |   | h ← max(h, 1 + hei(b))
```

$hei(a0) = 3$ $\quad hei(a3) = 2$

$hei(a1) = 1$ $\quad hei(a4) = 4$

$hei(a2) = 3$ $\quad hei(a5) = 1$

I*b*. *Функции проверки простоты массива ma*

```
leaves(ma) := | nu ← 0
              | for a ∈ ma
              |   | nu ← nu + IsArray(a)
              | nu = 0
```

$leaves(a0) = 0$ $\quad leaves(a3) = 0$

$leaves(a1) = 1$ $\quad leaves(a4) = 0$

$leaves(a2) = 0$ $\quad leaves(a5) = 1$

I*c*. *Рекурсивная функции создания по массиву ma вектора всех его простых подмассивов.*

```
vecsim(ma) := | ve ← if(leaves(ma), return [ma], [0])
              | for b ∈ ma
              |   | if IsArray(b)
              |   |   | ve ← stack(ve, if(leaves(b), [b], vecsim(b)))
              | submatrix(ve, 1, last(ve), 0, 0)
```

$$vecsim(a0)^{T} = \begin{bmatrix} [7\ \ 1] & \begin{bmatrix} 2 \\ 6 \end{bmatrix} & [7\ \ 1\ \ 1\ \ 6] & \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} & [7\ \ 1] \end{bmatrix} \qquad vecsim(a1) = [[5]]$$

Фрагмент 2 (2–3)

$$vecsim(a3)^T = [[1]\ [2]] \qquad vecsim(a4)^T = [[1]\ [2]] \qquad vecsim(a5)^T = \left[\begin{bmatrix}1 & 1 & 2\\ 2 & 1 & 3\end{bmatrix}\right]$$

I*d. Функции создания по массиву та вектора всех его простых различных подмассивов* (повторяющиеся подмассивы в вектор входят по одному разу).

$$vecsims(ma) := \left\| \begin{array}{l} ve \leftarrow vecsim(ma) \\ \begin{bmatrix} ot_0 & k \end{bmatrix} \leftarrow \begin{bmatrix} ve_0 & 1 \end{bmatrix} \\ \text{if rows}(ve) > 1 \\ \quad \left\| \begin{array}{l} \text{for } p \in 1..\text{last}(ve) \\ \quad \left\| \begin{array}{l} \text{if } \left(\prod_{i=0}^{p-1} (ve_i \neq ve_p) \right) \\ \quad \left\| \begin{bmatrix} ot_k & k \end{bmatrix} \leftarrow \begin{bmatrix} ve_p & k+1 \end{bmatrix} \right. \end{array} \right. \end{array} \right. \\ ot \end{array} \right.$$

Функция *vecsims* - не рекурсивна, функция *vecsim* - рекурсивна.

$$vecsims(a0)^T = \left[[7\ \ 1]\ \begin{bmatrix}2\\6\end{bmatrix}\ [7\ \ 1\ \ 1\ \ 6]\ \begin{bmatrix}7 & 2\\ 4 & 5\end{bmatrix} \right] \qquad vecsims(a1) = [[5]]$$

$$vecsims(a3)^T = [[1]\ [2]] \qquad vecsims(a4)^T = [[1]\ [2]] \qquad vecsims(a5)^T = \left[\begin{bmatrix}1 & 1 & 2\\ 2 & 1 & 3\end{bmatrix}\right]$$

II. Генерирование случайных гнездовых массивов

II*a. Рекурсивная функция rtree.* По *rtree* генерируется случайный гнездовой массив со случайными подмассивами (случайные размеры, случайные целочисленные элементы, случайная глубина вложенности). Все случайные параметры легко модифицируются.

$$rtree(p) := \left\| \begin{array}{l} [n\ \ m] \leftarrow [\text{floor}(\text{rnd}(3))\ \ \text{floor}(\text{rnd}(3))] \\ \text{for } i \in 0..n \\ \quad \left\| \begin{array}{l} \text{for } j \in 0..m \\ \quad \left\| B_{i,j} \leftarrow \text{if}(\text{rnd}(1) < p, rtree(p), \text{floor}(\text{rnd}(9))) \right. \end{array} \right. \\ B \end{array} \right.$$

II*b. Головная функция setree.* Функция *setree* обрабатывает возможные "ошибки", которые могут образоваться в *rtree* при прямом обращении к этой функции. В частности высота генерируемых массивов в *setree* ограничивается числом 5.

$$setree(p) := \left\| \begin{array}{l} \text{try} \\ \quad \left\| \begin{array}{l} B \leftarrow rtree(p) \\ \text{while } hei(B) > 5 \\ \quad \left\| B \leftarrow rtree(p) \right. \end{array} \right. \\ \text{on error} \\ \quad \left\| setree(p) \right. \end{array} \right.$$

$$y := \text{Seed}(1771313129) \qquad z := setree(0.2)$$

$$z = \left[1 \quad \begin{bmatrix} 7 & 6 \\ 8 & \begin{bmatrix}8\\2\\0\end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \begin{bmatrix} 5 & & 5 \\ 7 & & [3\ \ 5\ \ 3] \\ [8] & & 7 \end{bmatrix} & \begin{matrix} 0 \\ \\ \end{matrix} & \begin{matrix} 2 \\ 7 \end{matrix} \end{bmatrix} \right] \qquad hei(z) = 4$$

Фрагмент 2 (3–4)

III. Функции для тестирования программ

```
testf(n,g,h,P) := | y ← Seed(73190595)
                  | p ← 0.2
                  | for k ∈ 1..n
                  |   | ma ← setree(p)
                  |   | if g(ma,P) ≠ h(ma,P)
                  |   |   | return "нет"
                  | "да"
```

Функция *testf*(*n*,*g*,*h*,*P*) для сравнения результатов выполнения функций *g* и *h* на каждом из *n* последовательно генерируемых случайных гнездовых массивах *ma* при наличии в решаемой задаче функции-уточнителя *P*.

```
testftime(n,g,P) := | y ← Seed(73190595)
                    | p ← 0.2
                    | t ← time(1)
                    | for k ∈ 1..n
                    |   | ma ← setree(p)
                    |   | ta ← g(ma,P)
                    | time(1) − t
```

Функция *testftime*(*n*,*g*,*P*) для вычисления времени выполнения функции *g* на *n* последовательно генерируемых случайных гнездовых массивах *ma* при наличии в решаемой задаче функции-уточнителя *P*.

Тест 1. *Функция testf (n, g, h, P) для n сравнений результатов выполнения функций g и h* при наличии фиксированной логической функции-параметра *P*. Пусть для гнездовых массивов *ma* решается некоторая задача *Z* (*ma*, *P*). Если имеются две функции *g* (*ma*, *P*) и *h* (*ma*, *P*) решения *Z*, то функция *testf* (*n*, *g*, *h*, *P*) тестирует *g* и *h* на совпадение результатов вычислений на каждом из *n* последовательно генерируемых случайных гнездовых массивов *ma*. Если во всех опытах *g* (*ma*, *P*) = *h* (*ma*, *P*), то возвращается сообщение «*да*», в противном случае — «*нет*».

Тест 2. *Функция testftime (n, g, P) для вычисления времени выполнения функции g n раз* при наличии фиксированной логической функции-параметра *P*. Пусть для гнездовых массивов *ma* решается некоторая задача *Z* (*ma*, *P*). Если имеется функция *g* (*ma*, *P*) решения этой задачи, то *testftime* (*n*, *g*, *P*) возвращает время выполнения *g* при *n* вычислениях, то есть на *n* последовательно генерируемых случайных гнездовых массивах *ma*.

IV. Создание и тестирование параметрических функций. На фрагменте 3 представлены следующие функции для решения задач *A*1–*A*2, *B*1–*B*2, *C*1–*C*3: *A*1 — любая из функций *Aone*1 и *Aone*2; *A*2 — любая из функций *Aall*1–*Aall*3; *B*1 — любая из функций *Bone*1–*Bone*3; *B*2 — любая из функций *Ball*1–*Ball*2; *C*1 — любая из функций *Cone*1–*Cone*2; *C*2 — любая из функций *Call*1–*Call*2; *C*3 — любая из функций *Callv*1–*Callv*2.

Фрагмент 3 (1–10)

Параметрические функции для обработки гнездовых массивов

Ниже размещена ресурсная ссылка на файл *frag2_preface-funcs.mcdx* фрагмента 2. Она делает доступными в текущем документе определения всех переменных и всех функций из этого файла.

Включить << D:\0\frag2_preface-funcs.mcdx

I. Задачи *A* $n := 100000$

I*a*. Задачу *A*1 можно решать любой из функций *Aone*1 и *Aone*2 (находится простой подмассив *sa* из гнездового массива *ma*, для которого логическая функция *L* удовлетворяет условию $L(sa)=1$ (истина). При неудаче возвращается слово «нет»).

a.

```
Aone1(ma, L) := | ot ← "нет"
                | if leaves(ma)
                |   | if(L(ma), ot ← ma, 0)
                | else
                |   | for a ∈ ma
                |   |   | if IsArray(a)
                |   |   |   | if leaves(a)
                |   |   |   |   | if(L(a), return a, 0)
                |   |   |   | else
                |   |   |   |   | y ← Aone1(a, L)
                |   |   |   |   | if(y ≠ "нет", return y, 0)
                | ot
```

*Aone*1 - рекурсивна; *leaves* - не рекурсивна.

b.

```
Aone2(ma, L) := | for k ∈ 0..last(ve ← vecsim(ma))
                |   | if(L(ve_k), return ve_k, "нет")
```

*Aone*2 - не рекурсивна; *vecsim* - рекурсивна.

I*b*. Задачу *A*2 можно решать любой из представленных ниже функций *Aall*1-*Aall*3 (находится вектор всех простых подмассивов *sa* из *ma*, на которых *P*(*sa*)=1).

a.

```
Aall1(ma, L) := | ot ← 0
                | for k ∈ 0..last(ve ← vecsim(ma))
                |   | if L(ve_k)
                |   |   | ot ← stack(ot, [ve_k])
                | if(ot = 0, ["нет"], submatrix(ot, 1, last(ot), 0, 0))
```

*Aall*1-не рекурсивна; *vecsim* - рекурсивна.

Фрагмент 3 (2–10)

b. $Aall2(ma,L) :=$

```
ve ← ["нет"]
if leaves(ma)
    if(L(ma), return [ma], 0)
for b ∈ ma
    if IsArray(b)
        if leaves(b)
            if(L(b), ve ← stack(ve, [b]), 0)
        else
            y ← Aall2(b, L)
            if(y ≠ ["нет"], ve ← stack(ve, y), 0)
if(ve = ["нет"], ve, submatrix(ve, 1, last(ve), 0, 0))
```

Aall2 - рекурсивна; *leaves* - не рекурсивна.

c. $recl(ma,L,ve) :=$

```
for b ∈ ma
    if IsArray(b)
        if leaves(b)
            if(L(b), ve ← stack(ve, [b]), 0)
        else
            y ← recl(b, L, ["нет"])
            if(y ≠ ["нет"], ve ← stack(ve, y), 0)
if(ve = ["нет"], ve, submatrix(ve, 1, last(ve), 0, 0))
```

$Aall3(ma,L) :=$

```
if leaves(ma)
    if(L(ma), return [ma], 0)
ve ← recl(ma, L, ["нет"])
```

recl - рекурсивна; *leaves* - не рекурсивна; *Aall3* - головная программа.

I*c*. Контрольные вычисления. Вычисления по функциям *Aone*1-*Aone2*, *Aall*1-*Aall3* при конкретных заданиях логической функции-параметра *L*.

1) Существуют элементы, равные *b*. $b := 6$

$L1(x) :=$

```
for i ∈ 0..rows(x)−1
    for j ∈ 0..cols(x)−1
        if(x[i,j] = b, return 1, 0)
```

$$Aone1(a0,L1)=\begin{bmatrix}2\\6\end{bmatrix} \quad Aone2(a0,L1)=\begin{bmatrix}2\\6\end{bmatrix}$$

$$Aall1(a0,L1)=\begin{bmatrix}\begin{bmatrix}2\\6\end{bmatrix}\\ \begin{bmatrix}7&1&1&6\end{bmatrix}\end{bmatrix} \quad Aall2(a0,L1)=\begin{bmatrix}\begin{bmatrix}2\\6\end{bmatrix}\\ \begin{bmatrix}7&1&1&6\end{bmatrix}\end{bmatrix} \quad Aall3(a0,L1)=\begin{bmatrix}\begin{bmatrix}2\\6\end{bmatrix}\\ \begin{bmatrix}7&1&1&6\end{bmatrix}\end{bmatrix}$$

2) Сумма элементов равна *b*. $b := 18$

Фрагмент 3 (3–10)

$$L2(x) := \left(\sum_{i=0}^{\mathrm{rows}(x)-1} \sum_{j=0}^{\mathrm{cols}(x)-1} x_{i,j} \right) = b \qquad Aone1(a0, L2) = \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \qquad Aone2(a0, L2) = \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix}$$

$$Aall1(a0, L2) = \left[\begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \right] \qquad Aall2(a0, L2) = \left[\begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \right] \qquad Aall3(a0, L2) = \left[\begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \right]$$

3) Столбцы упорядочены по неубыванию.

$$L3(x) := \sum_{k=0}^{\mathrm{cols}(x)-1} \left(x^{\langle k \rangle} = \mathrm{sort}\left(x^{\langle k \rangle}\right) \right) = \mathrm{cols}(x) \qquad \begin{aligned} &Aone1(a0, L3) = \begin{bmatrix} 7 & 1 \end{bmatrix} \\ &Aone2(a0, L3) = \begin{bmatrix} 7 & 1 \end{bmatrix} \end{aligned}$$

$$Aall1(a0, L3) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \qquad Aall2(a0, L3) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \qquad Aall3(a0, L3) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix}$$

4) Ранг матрицы полный.

$$L4(x) := (\mathrm{rank}(x) = min(\mathrm{rows}(x), \mathrm{cols}(x))) \qquad \begin{aligned} &Aone1(a0, L4) = \begin{bmatrix} 7 & 1 \end{bmatrix} \\ &Aone2(a0, L4) = \begin{bmatrix} 7 & 1 \end{bmatrix} \end{aligned}$$

$$Aall1(a0, L4) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \qquad Aall2(a0, L4) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \qquad Aall3(a0, L4) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix}$$

I*d*. Тестирование на совпадение результатов. Тестируются функции *Aone*1-*Aone*2 и *Aall*1-*Aall*3 при *n* вычислениях на случайных массивах при конкретных заданиях логической функции-параметра *L* (см. I*c*). Во всех случаях проверка проводится при *n*=100000.

a. $testf(n, Aone1, Aone2, L1) =$ “да” $\qquad testf(n, Aone1, Aone2, L3) =$ “да”
$testf(n, Aone1, Aone2, L2) =$ “да” $\qquad testf(n, Aone1, Aone2, L4) =$ “да”

b. "Векторное" тестирование сразу для нескольких функций-параметров.

$$VL := \begin{bmatrix} L1 & L2 & L3 & L4 \end{bmatrix}^{\mathrm{T}} \qquad \overrightarrow{testf(n, Aone1, Aone2, VL)}^{\mathrm{T}} = \begin{bmatrix} \text{“да”} & \text{“да”} & \text{“да”} & \text{“да”} \end{bmatrix}$$

$$\overrightarrow{testf(n, Aall1, Aall2, VL)}^{\mathrm{T}} = \begin{bmatrix} \text{“да”} & \text{“да”} & \text{“да”} & \text{“да”} \end{bmatrix}$$

$$\overrightarrow{testf(n, Aall1, Aall3, VL)}^{\mathrm{T}} = \begin{bmatrix} \text{“да”} & \text{“да”} & \text{“да”} & \text{“да”} \end{bmatrix}$$

$$\overrightarrow{testf(n, Aall2, Aall3, VL)}^{\mathrm{T}} = \begin{bmatrix} \text{“да”} & \text{“да”} & \text{“да”} & \text{“да”} \end{bmatrix}$$

I*e*. Тестирование на время выполнения. Тестируются функции *Aone*1-*Aone*2, *Aall*1-*Aall*3 при условиях I*d*.

Фрагмент 3 (4–10)

$$\overrightarrow{testftime(n, Aone1, VL)}^{T} = [6.74\ \ 7.09\ \ 6.77\ \ 6.72]$$

$$\overrightarrow{testftime(n, Aone2, VL)}^{T} = [7.58\ \ 7.53\ \ 7.48\ \ 7.67]$$

$$\overrightarrow{testftime(n, Aall1, VL)}^{T} = [7.84\ \ 7.5\ \ 7.9\ \ 8.29]$$

$$\overrightarrow{testftime(n, Aall2, VL)}^{T} = [7.69\ \ 7.39\ \ 7.7\ \ 7.81]$$

$$\overrightarrow{testftime(n, Aall3, VL)}^{T} = [7.01\ \ 6.89\ \ 7.39\ \ 7.64]$$

Выводы. Функция *Aone*1 на проверенных задачах выполняется несколько быстрее *функции Aone*2. Время выполнения *Aall*1, *Aall*2 и *Aall*3 на этих задачах приблизительно одинаковое. Незначительное замедление демонстрирует функция *Aall*1.

II. Задачи *B*

II*a*. Задачу *B*1 можно решать любой из *следующих функций Bone*1*-Bone*3 (ищется простой подмассив sa из ma такой, что *P*(*sa*,*x*)=1 при любом простом подмассиве *x* из *ma*).

```
Bone1(ma,P) := | vl ← vecsim(ma)
               | for k ∈ 0..last(vl)
               |   | if rows(vl) = Σ_{s=0}^{last(vl)} P(vl_k, vl_s)
               |   |   | return vl_k
               | "нет"
```

*Bone*1*-Bone*3
не рекурсивны;
vecsim - рекурсивна.

```
Bone2(ma,P) := | vl ← vecsim(ma)
               | for k ∈ 0..last(vl)
               |   | if Π_{s=0}^{last(vl)} P(vl_k, vl_s)
               |   |   | return vl_k
               | "нет"
```

```
Bone3(ma,P) := | vl ← vecsim(ma)
               | for k ∈ 0..last(vl)
               |   | if( Π_{s=0}^{last(vl)} P(vl_k, vl_s), return vl_k, "нет")
```

II*b*. Задачу *B*2 можно решать любой из следующих функций *Ball*1*-Ball*2 (ищется вектор простых подмассивов sa из ma таких, что *P*(*sa*,*x*)=1 при любом простом подмассиве *x* из *ma*).

Фрагмент 3 (5–10)

$$Ball1(ma,P):=\left\|\begin{array}{l}[vl\ \ p\ \ ot]\leftarrow[vecsim(ma)\ \ 0\ \ \text{“нет”}]\\ \text{for } k\in 0..\text{last}(vl)\\ \quad\left\|\begin{array}{l}\text{if rows}(vl)=\sum\limits_{s=0}^{\text{last}(vl)}P\left(vl_k,vl_s\right)\\ \quad\left\|\left[ot_p\leftarrow vl_k\ \ p\leftarrow p+1\right]\right.\end{array}\right.\\ ot\end{array}\right.$$

Ball1 и *Ball2* - не рекурсивны; *vecsim* - рекурсивна

$$Ball2(ma,P):=\left\|\begin{array}{l}[vl\ \ p\ \ ot]\leftarrow[vecsim(ma)\ \ 0\ \ \text{“нет”}]\\ \text{for } k\in 0..\text{last}(vl)\\ \quad\left\|\begin{array}{l}\text{if }\prod\limits_{s=0}^{\text{last}(vl)}P\left(vl_k,vl_s\right)\\ \quad\left\|\left[ot_p\leftarrow vl_k\ \ p\leftarrow p+1\right]\right.\end{array}\right.\\ ot\end{array}\right.$$

II*c*. Контрольные вычисления. Вычисления по функциям *Bone1*-*Bone4*, *Ball1* и *Ball2* при конкретных заданиях логической функции-параметра *P*.

1) Количество элементов в *sa* равно количеству элементов в *x* (=,<=,>=,<,>):

$P1(sa,x):=\text{rows}(sa)\cdot\text{cols}(sa)=\text{rows}(x)\cdot\text{cols}(x)$

$P2(sa,x):=\text{rows}(sa)\cdot\text{cols}(sa)\le\text{rows}(x)\cdot\text{cols}(x)$

$P3(sa,x):=\text{rows}(sa)\cdot\text{cols}(sa)\ge\text{rows}(x)\cdot\text{cols}(x)$

$P4(sa,x):=\text{rows}(sa)\cdot\text{cols}(sa)<\text{rows}(x)\cdot\text{cols}(x)$

$P5(sa,x):=\text{rows}(sa)\cdot\text{cols}(sa)>\text{rows}(x)\cdot\text{cols}(x)$

$Bone1(a0,P1)=\text{“нет”}$ $\quad Bone1(a0,P2)=[7\ \ 1]$ $\quad Bone1(a0,P3)=[7\ \ 1\ \ 1\ \ 6]$

$Bone1(a0,P4)=\text{“нет”}$ $\quad Bone1(a0,P5)=\text{“нет”}$

$Ball1(a0,P1)=\text{“нет”}$

$Ball1(a0,P4)=\text{“нет”}$

$$Ball1(a0,P2)=\begin{bmatrix}[7\ \ 1]\\ [2]\\ [6]\\ [7\ \ 1]\end{bmatrix}\qquad Ball1(a0,P3)=\begin{bmatrix}[7\ \ 1\ \ 1\ \ 6]\\ \begin{bmatrix}7&2\\4&5\end{bmatrix}\end{bmatrix}$$

2) Сумма элементов в *sa* равна сумме элементов в *x* (=,<=,>=):

$$P6(sa,x):=\sum_{i=0}^{\text{rows}(sa)-1}\sum_{j=0}^{\text{cols}(sa)-1}sa_{i,j}=\sum_{i=0}^{\text{rows}(x)-1}\sum_{j=0}^{\text{cols}(x)-1}x_{i,j}$$

$$P7(sa,x):=\sum_{i=0}^{\text{rows}(sa)-1}\sum_{j=0}^{\text{cols}(sa)-1}sa_{i,j}\le\sum_{i=0}^{\text{rows}(x)-1}\sum_{j=0}^{\text{cols}(x)-1}x_{i,j}$$

$$P8(sa,x):=\sum_{i=0}^{\text{rows}(sa)-1}\sum_{j=0}^{\text{cols}(sa)-1}sa_{i,j}\ge\sum_{i=0}^{\text{rows}(x)-1}\sum_{j=0}^{\text{cols}(x)-1}x_{i,j}$$

$Bone1(a0,P6)=\text{“нет”}$ $\quad Bone1(a0,P7)=[7\ \ 1]$ $\quad Bone1(a0,P8)=\begin{bmatrix}7&2\\4&5\end{bmatrix}$

Фрагмент 3 (6–10)

$Ball1(a0,P6) = $ "нет" $\quad Ball1(a0,P7) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \quad Ball1(a0,P8) = \begin{bmatrix} \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \end{bmatrix}$

3) Максимальный элемент в *sa* равен максимальному элемента в *x* (=,<=,>=):

$P9(sa,x) := \max(sa) = \max(x)$
$P10(sa,x) := \max(sa) \le \max(x)$
$P11(sa,x) := \max(sa) \ge \max(x)$

$Bone1(a0,P9) = $ "нет" $\quad Bone1(a0,P10) = \begin{bmatrix} 2 \\ 6 \end{bmatrix} \quad Bone1(a0,P11) = \begin{bmatrix} 7 & 1 \end{bmatrix}$

$Ball1(a0,P9) = $ "нет" $\quad Ball1(a0,P10) = \begin{bmatrix} \begin{bmatrix} 2 \\ 6 \end{bmatrix} \end{bmatrix} \quad Ball1(a0,P11) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix}$

4) Минимальный элемент в *sa* равен минимальному элементу в *x* (=,<=,>=):

$P12(sa,x) := min(sa) = \min(x)$
$P13(sa,x) := \min(sa) \le \min(x)$
$P14(sa,x) := \min(sa) \ge \min(x)$

$Bone1(a0,P12) = $ "нет" $\quad Bone1(a0,P13) = \begin{bmatrix} 7 & 1 \end{bmatrix} \quad Bone1(a0,P14) = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$

$Ball1(a0,P12) = $ "нет" $\quad Ball1(a0,P13) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \quad Ball1(a0,P14) = \begin{bmatrix} \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \end{bmatrix}$

5) Ранг матрицы *sa* меньше ранга матрицы *x* (<,<=,>=):

$P15(sa,x) := \mathrm{rank}(sa) < \mathrm{rank}(x)$
$P16(sa,x) := \mathrm{rank}(sa) \le \mathrm{rank}(x)$
$P17(sa,x) := \mathrm{rank}(sa) \ge \mathrm{rank}(x)$

$Bone1(a0,P15) = $ "нет" $\quad Bone1(a0,P16) = \begin{bmatrix} 7 & 1 \end{bmatrix} \quad Bone1(a0,P17) = \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix}$

$Ball1(a0,P15) = $ "нет" $\quad Ball1(a0,P16) = \begin{bmatrix} \begin{bmatrix} 7 & 1 \end{bmatrix} \\ \begin{bmatrix} 2 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 7 & 1 \end{bmatrix} \end{bmatrix} \quad Ball1(a0,P17) = \begin{bmatrix} \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \end{bmatrix}$

II*d*. Тестирование на совпадение результатов. Тестируются функции *Bone*1-*Bone*3 и *Ball*1-*Ball*2 при *n* вычислениях на случайных массивах при конкретных заданиях логической функции-параметра *P* (см. II*c*). Во всех случаях проверка проводится при *n*=100000.

"Векторное" тестирование сразу для нескольких функций-параметров.

$VP := \begin{bmatrix} P1 & P2 & P3 & P4 & P5 & P6 & P7 & P8 & P9 \end{bmatrix}^{\mathrm{T}}$

Фрагмент 3 (7–10)

$$WP := [P10\ \ P11\ \ P12\ \ P13\ \ P14\ \ P15\ \ P16\ \ P17]^{T}$$

a. $\overrightarrow{testf(n, Bone1, Bone2, VP)}^{T} = [\text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}]$

$\overrightarrow{testf(n, Bone1, Bone3, VP)}^{T} = [\text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}]$

$\overrightarrow{testf(n, Bone2, Bone3, VP)}^{T} = [\text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}]$

b. $\overrightarrow{testf(n, Ball1, Ball2, VP)}^{T} = [\text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}\ \text{“да”}]$

II*e*. Тестирование на время выполнения. Тестируются функции *Bone*1-*Bone*4 и *Ball*1-*Ball*2 при условиях II*d*.

a. $\overrightarrow{testftime(n, Bone1, VP)}^{T} = [7.57\ \ 7.41\ \ 7.41\ \ 7.64\ \ 7.5\ \ 8.65\ \ 8.36\ \ 8.41\ \ 7.48]$

$\overrightarrow{testftime(n, Bone2, VP)}^{T} = [7.59\ \ 7.4\ \ 7.43\ \ 7.65\ \ 7.48\ \ 8.59\ \ 8.26\ \ 8.29\ \ 7.54]$

$\overrightarrow{testftime(n, Bone3, VP)}^{T} = [7.59\ \ 7.47\ \ 7.46\ \ 7.6\ \ 7.57\ \ 8.77\ \ 8.41\ \ 8.61\ \ 7.66]$

b. $\overrightarrow{testftime(n, Ball1, VP)}^{T} = [7.71\ \ 7.6\ \ 7.69\ \ 7.49\ \ 7.51\ \ 8.9\ \ 8.88\ \ 8.94\ \ 7.67]$

$\overrightarrow{testftime(n, Ball2, VP)}^{T} = [7.63\ \ 6.97\ \ 6.82\ \ 6.58\ \ 6.57\ \ 7.76\ \ 7.88\ \ 7.82\ \ 6.69]$

Выводы. Функция $Bone2$ на проверенных задачах *P* выполняется несколько быстрее функций $Bone1$ и $Bone3$. Функция $Ball2$ на этих задачах *P* выполняется несколько быстрее функции $Ball1$.

III. Задачи *C*

III*a*. Задачу *C*1 можно решать любой из следующих функций *Cone*1-*Cone*2 (ищется простой подмассив sa из ma такой, что *P*(*sa*,*x*)=1 при некотором *x* из *ma* ($x \neq sa$)).

```
Cone1(ma,P) := | vl ← vecsim(ma)
               | for p ∈ 0..last(vl)
               |   | for q ∈ 0..last(vl)
               |   |   | if (vl_p ≠ vl_q) ∧ P(vl_p, vl_q)
               |   |   |   | return vl_p
               | "нет"
```

*Cone*1-*Cone*2 - не рекурсивны; *vecsim* - рекурсивна

$Cone1(a0, P1) = [7\ \ 1]$

$Cone1(a0, P5) = [7\ \ 1\ \ 1\ \ 6]$

```
Cone2(ma,P) := | vl ← vecsim(ma)
               | for p ∈ 0..last(vl)
               |   | for q ∈ 0..last(vl)
               |   |   | if((vl_p ≠ vl_q) ∧ P(vl_p, vl_q), return vl_p, "нет")
```

Фрагмент 3 (8–10)

III*b*. Задачу *C2* можно решать любой из функций *Call1*-*Call2* (ищется вектор всех простых подмассивов sa из ma такой, что *P*(*sa*,*x*)=1 при некотором *x* из *ma* ($x \notin sa$)).

```
Call1(ma,P) := | [vl  k  ot] ← [vecsim(ma)  0  "нет"]
               | for p ∈ 0..last(vl)
               |   if  Σ_{s=0}^{last(vl)} ((vl_p ≠ vl_s) ∧ P(vl_p, vl_s)) ≠ 0
               |     [ot_k  k] ← [vl_p  k+1]
               | ot
```

Call1, *Call2* - не рекурсивны;
vecsim - рекурсивна.

Подмассив [7 1] входит 2 раза.

```
Call2(ma,P) := | [vl  k  ot] ← [vecsim(ma)  0  "нет"]
               | for p ∈ 0..last(vl)
               |   for q ∈ 0..last(vl)
               |     if (vl_p ≠ vl_q) ∧ P(vl_p, vl_q)
               |       [ot_k  k] ← [vl_p  k+1]
               |       break
               | ot
```

$$Call2(a0,P1)=\begin{bmatrix} [7\ 1] \\ [2] \\ [6] \\ [7\ 1\ 1\ 6] \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \\ [7\ 1] \end{bmatrix}$$

$$Call2(a0,P1)=Call2(a0,P1)=1$$

III*c*. *Задачу C3* можно решать любой из функций *Callv1*-*Callv2* (ищется вектор всех неповторяющихся простых подмассивов sa из ma такой, что *P*(*sa*,*x*)=1 при некотором *x* из *ma* ($x \notin sa$)).

```
Callv1(ma,P) := | [vl  k  ot] ← [vecsim(ma)  0  "нет"]
                | for p ∈ 0..last(vl)
                |   for q ∈ 0..last(vl)
                |     if (vl_p ≠ vl_q) ∧ P(vl_p, vl_q)
                |       if k = 0
                |         [ot_k  k] ← [vl_p  k+1]
                |       else if  Σ_{s=0}^{k-1} (ot_s = vl_p) = 0
                |         [ot_k  k] ← [vl_p  k+1]
                |       break
                | ot
```

Callv1, *Callv2* - не рек.,
vecsim, *vecsims* - рек.

$$Callv1(a0,P1)=\begin{bmatrix} [7\ 1] \\ [2] \\ [6] \\ [7\ 1\ 1\ 6] \\ \begin{bmatrix} 7 & 2 \\ 4 & 5 \end{bmatrix} \end{bmatrix}$$

$$Callv1(a2,P1)=\text{“нет”}$$

$$Callv1(a4,P1)=\begin{bmatrix} [1] \\ [2] \end{bmatrix}$$

$$Callv1(a3,P1)=\begin{bmatrix} [1] \\ [2] \end{bmatrix}$$

Фрагмент 3 (9–10)

```
Callv2(ma,P) := | [vl  k  ot] ← [vecsims(ma)  0  "нет"]
                | for p ∈ 0..last(vl)
                |   for q ∈ 0..last(vl)
                |     if (vl_p ≠ vl_q) ∧ P(vl_p, vl_q)
                |       [ot_k  k] ← [vl_p  k+1]
                |       break
                | ot
```

$$Callv2(a0,P1)=\begin{bmatrix}\begin{bmatrix}7 & 1\end{bmatrix}\\ \begin{bmatrix}2\\6\end{bmatrix}\\ \begin{bmatrix}7 & 1 & 1 & 6\end{bmatrix}\\ \begin{bmatrix}7 & 2\\4 & 5\end{bmatrix}\end{bmatrix}$$

III*d*. Контрольные вычисления. Вычисления по функциям *Cone*1-*Cone*2, *Call*1-*Call*2 и *Callv*1-*Callv*2 при конкретных заданиях логической функции-параметра *P* (см. II*c*).

a.

$Cone1(a0,P1)=[7\ \ 1]$ $\qquad$ $Cone2(a0,P1)=[7\ \ 1]$

$Cone1(a1,P6)=\text{“нет”}$ $\qquad$ $Cone2(a1,P2)=\text{“нет”}$

$Cone1(a2,P9)=\text{“нет”}$ $\qquad$ $Cone2(a2,P9)=\text{“нет”}$

$Cone1(a3,P17)=[1]$ $\qquad$ $Cone2(a3,P17)=[1]$

$Cone1(a3,P10)=[1]$ $\qquad$ $Cone2(a3,P10)=[1]$

$Cone1(a0,P5)=[7\ \ 1\ \ 1\ \ 6]$ $\qquad$ $Cone2(a0,P5)=[7\ \ 1\ \ 1\ \ 6]$

b.

$Call1(a1,P6)=\text{“нет”}$ $\qquad$ $Call2(a1,P2)=\text{“нет”}$

$Call1(a3,P17)=\begin{bmatrix}[1]\\ [2]\end{bmatrix}$ $\qquad$ $Call2(a3,P17)=\begin{bmatrix}[1]\\ [2]\end{bmatrix}$

$Call1(a3,P10)=[[1]]$ $\qquad$ $Call2(a3,P10)=[[1]]$

$Call1(a0,P5)^{T}=\left[\begin{bmatrix}7 & 1 & 1 & 6\end{bmatrix}\ \begin{bmatrix}7 & 2\\4 & 5\end{bmatrix}\right]$ $\qquad$ $Call2(a0,P5)^{T}=\left[\begin{bmatrix}7 & 1 & 1 & 6\end{bmatrix}\ \begin{bmatrix}7 & 2\\4 & 5\end{bmatrix}\right]$

$Call1(a0,P4)^{T}=\left[\begin{bmatrix}7 & 1\end{bmatrix}\ \begin{bmatrix}2\\6\end{bmatrix}\ \begin{bmatrix}7 & 1\end{bmatrix}\right]$ $\qquad$ $Call2(a0,P4)^{T}=\left[\begin{bmatrix}7 & 1\end{bmatrix}\ \begin{bmatrix}2\\6\end{bmatrix}\ \begin{bmatrix}7 & 1\end{bmatrix}\right]$

c.

$Callv1(a0,P5)^{T}=\left[\begin{bmatrix}7 & 1 & 1 & 6\end{bmatrix}\ \begin{bmatrix}7 & 2\\4 & 5\end{bmatrix}\right]$ $\qquad$ $Callv2(a0,P5)^{T}=\left[\begin{bmatrix}7 & 1 & 1 & 6\end{bmatrix}\ \begin{bmatrix}7 & 2\\4 & 5\end{bmatrix}\right]$

$Callv1(a0,P4)^{T}=\left[\begin{bmatrix}7 & 1\end{bmatrix}\ \begin{bmatrix}2\\6\end{bmatrix}\right]$ $\qquad$ $Callv2(a0,P4)^{T}=\left[\begin{bmatrix}7 & 1\end{bmatrix}\ \begin{bmatrix}2\\6\end{bmatrix}\right]$

III*e*. Тестирование на совпадение результатов. Тестируются функции *Cone*1-*Cone*2, *Call*1-*Call*2 и *Callv*1-*Callv*2 при *n* вычислениях на случайных массивах при конкретных заданиях логической функции-параметра *P* (см. II*c*). Во всех случаях реализуется для *n*=100000.

$$\overrightarrow{testf(n,Cone1,Cone2,VP)}^{T}=[\text{“да” “да” “да” “да” “да” “да” “да” “да” “да”}]$$

$$\overrightarrow{testf(n,Cone1,Cone2,WP)}^{T}=[\text{“да” “да” “да” “да” “да” “да” “да” “да”}]$$

$$\overrightarrow{testf(n,Call1,Call2,VP)}^{T}=[\text{“да” “да” “да” “да” “да” “да” “да” “да” “да”}]$$

Фрагмент 3 (10–10)

$$\overrightarrow{testf(n, Call1, Call2, WP)}^{T} = [\text{“да” “да” “да” “да” “да” “да” “да” “да”}]$$

$$\overrightarrow{testf(n, Callv1, Callv2, VP)}^{T} = [\text{“да” “да” “да” “да” “да” “да” “да” “да” “да”}]$$

$$\overrightarrow{testf(n, Callv1, Callv2, WP)}^{T} = [\text{“да” “да” “да” “да” “да” “да” “да” “да”}]$$

III*f*. Тестирование на время выполнения. Тестируются функции *Cone*1-*Cone*2, *Call*1-*Call*2 и *Callv*1-*Callv*2 при условиях III*e*.

a. $\overrightarrow{testftime(n, Cone1, VP)}^{T} = [7.36\ \ 7.35\ \ 7.38\ \ 7.32\ \ 7.4\ \ 8.64\ \ 8.37\ \ 8.14\ \ 7.55]$

$\overrightarrow{testftime(n, Cone2, VP)}^{T} = [7.51\ \ 7.45\ \ 7.43\ \ 7.38\ \ 7.38\ \ 8.88\ \ 8.31\ \ 8.61\ \ 7.52]$

b. $\overrightarrow{testftime(n, Call1, VP)}^{T} = [7.81\ \ 7.93\ \ 7.76\ \ 7.72\ \ 7.76\ \ 8.96\ \ 9.08\ \ 9.05\ \ 7.8]$

$\overrightarrow{testftime(n, Call2, VP)}^{T} = [7.62\ \ 7.6\ \ 7.56\ \ 7.67\ \ 7.72\ \ 9\ \ 8.82\ \ 8.66\ \ 7.7]$

c. $\overrightarrow{testftime(n, Callv1, VP)}^{T} = [7.39\ \ 7.48\ \ 7.54\ \ 7.46\ \ 7.4\ \ 8.74\ \ 8.63\ \ 8.53\ \ 7.58]$

$\overrightarrow{testftime(n, Callv2, VP)}^{T} = [7.79\ \ 8.01\ \ 7.74\ \ 7.76\ \ 7.77\ \ 9.13\ \ 8.71\ \ 8.68\ \ 7.76]$

Выводы. Сравниваемые пары функций на проверенных задачах *P* выполняются приблизительно за равное время.

Для созданных функций проведены контрольные вычисления, выполнено тестирование на совпадение значений для пар функций, решающих одну и ту же задачу (*testf*), а также организовано тестирование на время выполнения функций в секундах (*testftime*). В каждом случае тестирование проводилось n = 100 000 раз на случайных гнездовых массивах *ma* при конкретных заданиях логической функции-параметра *P*. При тестировании использовалась операция векторизации по различным функциям *P*. Приводимое время выполнения функций является суммарным для n тестирований.

В заключение отметим возможные продолжения затронутой темы. В формулировках всех рассмотренных задач вместо поиска тех или иных подмассивов гнездовой матрицы *ma* можно было бы вести речь:

- о поиске их позиций в *ma*;
- о замещении найденных подмассивов *ma* какими-либо объектами (скалярами, строками, другими простыми или гнездовыми массивами).

Построение функций для решения этих задач может быть проведено по аналогии с соответствующими созданными функциями, показанными в данной статье, хотя на этом пути и возникает ряд нюансов. Впрочем, они носят рутинный характер и легко преодолимы.

Литература

1. *Григорьев С.Г.*, *Есаян А.Р.* Простой и обобщенный поиск элементов в гнездовых массивах и их замещение // Чебышевский сборник. Т. XVI. 2015. Вып. 3 (55). С. 460–478.

2. *Есаян А.Р., Добровольский Н.М.* Гнездовые массивы и рекурсия // Алгебра, теория чисел и дискретная геометрия. Современные проблемы и приложения: материалы XXIII международной конференции (г. Тула, 25–30 мая 2015 г.). Тула: Изд-во Тульского государственного педагогического университета им. Л.Н. Толстого, 2015. С. 319–321.

3. *Есаян А.Р., Добровольский Н.М.* Гнездовые массивы и рекурсия // Чебышевский сборник. Т. XVI. 2015. Вып. 3 (55). С. 479–495.

4. *Есаян А.Р., Якушин А.В.* Векторизация и гнездовые массивы // Алгебра, теория чисел и дискретная геометрия. Современные проблемы и приложения: материалы XXIII международной конференции (г. Тула, 25–30 мая 2015 г.). Тула: Изд-во Тульского государственного педагогического университета им. Л.Н. Толстого, 2015. С. 328–330.

5. *Есаян А.Р., Якушин А.В.* Векторизация и гнездовые массивы // Чебышевский сборник. Т. XVI. 2015. Вып. 3 (55). С. 496–509.

6. *Brent Maxfield P.E.* Essential PTC Mathcad Prime 3.0. A Guide for New and Current Users, New York, Academic Press is an imprint of Elsevier, Nov. 11, 2013. 563 p.

7. Hans Wessenlingh and Hans de Waard. Calculate & Communicate with Mathcad Prime 3.0, Delft Academic Press, The Netherlands, First edition 2014.

Literatura

1. *Grigor’ev S.G., Esayan A.R.* Prostoj i obobshhenny’j poisk e’lementov v gnezdovy’x massivax i ix zameshhenie // Cheby’shevskij sbornik. T. XVI. 2015. Vy’p. 3 (55). C. 460–478.

2. *Esayan A.R., Dobrovol’skij N.M.* Gnezdovy’e massivy’ i rekursiya // Algebra, teoriya chisel i diskretnaya geometriya. Sovremenny’e problemy’ i prilozheniya: materialy’ XXIII mezhdunarodnoj konferencii (g. Tula, 25–30 maya 2015 g.). Tula: Izd-vo Tul’skogo gosudarstvennogo pedagogicheskogo universiteta im. L.N. Tolstogo, 2015. S. 319–321.

3. *Esayan A.R., Dobrovol’skij N.M.* Gnezdovy’e massivy’ i rekursiya // Cheby’shevskij sbornik. T. XVI. 2015. Vy’p. 3 (55). S. 479–495.

4. *Esayan A.R., Yakushin A.V.* Vektorizaciya i gnezdovy’e massivy’ // Algebra, teoriya chisel i diskretnaya geometriya. Sovremenny’e problemy’ i prilozheniya: materialy’ XXIII mezhdunarodnoj konferencii (g. Tula, 25–30 maya 2015 g.). Tula: Izd-vo Tul’skogo gosudarstvennogo pedagogicheskogo universiteta im. L.N. Tolstogo, 2015. S. 328–330.

5. *Esayan A.R., Yakushin A.V.* Vektorizaciya i gnezdovy’e massivy’ // Cheby’shevskij sbornik. T. XVI. 2015. Vy’p. 3 (55). S. 496–509.

6. *Brent Maxfield P.E.* Essential PTC Mathcad Prime 3.0. A Guide for New and Current Users, New York, Academic Press is an imprint of Elsevier, Nov. 11, 2013. 563 p.

7. Nans Wessenlingh and Hans de Waard. Calculate & Communicate with Mathcad Prime 3.0, Delft Academic Press, The Netherlands, First edition 2014.

S.G. Grigoriev,
N.M. Dobrovolsky,
A.R. Yesayan

Parametric Functions for Processing Nested Arrays

In the article the authors propose the concept of processing nested arrays *M* by *F* custom functions with their implementation in the system *PTC Mathcad Prime.* The essence of the concept is that function *F* being created as one of its arguments should have secondary embedded or user-defined function *f*, working with simple arrays, numbers or lines. Fixing *f* while accessing *F*, we will solve the specific task of handling *M*. Moreover, if *f* is a built-in function, we can refer to *F* directly, but if not — then you need to pre-create a user-defined function *f*.

Keywords: computer science; nested arrays; information technologies; system of engineering and scientific computings *PTC Mathcad Prime*.